What is Claimed is:

1. A method for creating an algorithm module that can be used without change in a plurality of frameworks, the method comprising the steps of:

designing the algorithm module in a manner that renders the algorithm module reentrant within a preemptive environment;

coding each data access instruction of the algorithm module in a manner that renders the algorithm module and all of the data access instructions relocatable; and

providing a memory interface within the algorithm module that supports both design-time object instantiation and dynamic object instantiation.

2. The method of Claim 1, further comprising the steps of:

conforming to a run-time convention of a selected high level language;

characterizing a ROM-ability mode of the algorithm module;

prohibiting direct access to a peripheral device;

packaging the algorithm module in an archive which has a name that follows a uniform naming convention;

naming each algorithm header using a uniform naming convention; and

naming all external identifiers according to a uniform naming convention.

3. The method of Claim 2, further comprising the step of providing the algorithm module with an initialization function and with a finalization function

4. The method of Claim 3, further comprising the step of providing the algorithm module with a header file that supports multiple inclusions within a single source file.

5. The method of Claim 2, further comprising the step of providing a debug variable definition in a header of the algorithm module, wherein the debug variable definition uses the symbol _DEBUG.

6. The method of Claim 1, further comprising the step of implementing a trace interface as part of the algorithm module.

7. A method for converting an existing algorithm to an algorithm module that can be used without change in a plurality of frameworks, the method comprising the steps of:

providing a memory interface for the existing algorithm to form the algorithm module such that the algorithm module supports both design-time object instantiation and dynamic object instantiation;

revising the existing algorithm in a manner that renders the algorithm module reentrant within a preemptive environment; and

verifying that each data access instruction of the algorithm module is coded in a manner that renders the algorithm module and all of the data access instructions relocatable.

8. The method of Claim 7, further comprising the steps of:

characterizing a ROM-ability mode of the algorithm module;

prohibiting direct access to a peripheral device;

packaging the algorithm module in an archive which has a name that follows a uniform naming convention;

naming each algorithm header using a uniform naming convention; and

naming all external identifiers according to a uniform naming convention.

9. The method of Claim 8, further comprising the step of providing the algorithm module with an initialization function and with a finalization function.

10. The method of Claim 9, further comprising the step of providing the algorithm module with a header file that supports multiple inclusions within a single source file.

11. The method of Claim 10, further comprising the step of providing a debug variable definition in a header of the algorithm module, wherein the debug variable definition uses the symbol _DEBUG.